Copy No. 8

DEFENCE R&D DÉFENSE

PRODAT ENHANCEMENT AND REPAIR

C.Maxwell

PRIOR DATA SCIENCES LTD.

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

Contractor Report

DREA CR 1999-048

May 1999

Défense nationale National Defence

Canada

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

DREA CR 1999/048

PRODAT
ENHANCEMENT AND REPAIR

C. MAXWELL

PRIOR DATA SCIENCES LTD.
Queen's Court, Suite 601
5475 Spring Garden Rd.
Halifax, NS
B3J 3T2

Scientific Authority
I.A. Fraser

W7707-8-5997
Contract Number

May 1999

CONTRACTOR REPORT

Prepared for

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

ABSTRACT

This report details work performed on the DREA data acquisition and analysis program PRODAT from Nov 1998 to Jan 1999. A Harmonic Mean Normalizer (Cox and Pace, Proc. 30th Asilomar Conference, Nov 96), a grey scale expansion routine, and a facility for generating multiple DAT files from ping processing were added. Problems with short FFTs and colour-direction sonogram printing were corrected. Loss performance on the MAXION platform with newer versions of PRODAR was investigated.

RÉSUMÉ

Le présent rapport décrit en détail les travaux exécutés de novembre 1998 à janvier 1999 dans le cadre du programme d=analyse et d=acquisition de données PRODAT du CRDA. Un normaliseur de moyenne harmonique (Cox et Pace, Actes de la 30e conférence d=Asilomar, novembre 1996), une routine d=expansion d=échelle de gris et un moyen de générer de multiples fichiers DAT à partir du traitement d=impulsions ont été ajoutés. Les problèmes d'impression des sonogrammes couleurs et des TFR courtes ont été corrigés. La perte de performance sur la plate-forme MAXION avec les nouvelles versions de PRODAT a été étudiée.

Table of Contents

Introduction 1
Compiling PRODAT 1
Revision Control - CVS 1
Compilation 1
Modification 2
Other Features 2
Usage 3
Location of Executables 3
New Grammar Items 3
Usage of Multiple.DAT File Pings 3
Usage of Harmonic Mean Normalizer 4
Usage of Greyscale Expansion 4
Usage of Timeseries Display 4
Task 1: Work for Ocean Acoustics Group 5
Multi-DAT Pings 6
Implementation 7
Work with Windows NT Port 7
Prodat Front End 7
Task 2: Harmonic Mean Normalizer 8
Discussion On Averagers 9
Implementation 10
Task 3: Correct Problems with Short FFTs 11
Normalization 11
Colour Printouts 11
Task 4: Wider Grams 12
Miscellaneous Issues 13
MAXION Memory 13
GenCom/SASIO 13
BUFSIZ 14
Miscellaneous Fixes 14
Contractor Accounts 14
Conclusions 15
Bibliography 15
Appendices 16
A:Test26-98.set: PRODAT Parameter File for Task I 16
B:SET files testing Multi-DAT Pings 17
Apr24_1P. set 17
Apr24_1P-jp.set 17
C:SET Files for Harmonic Mean Normalizer Testing 18
harmm.set 18
D:SET Files for Tasks 3 and 4 18
task3.set 18
task4.set 19
task34.set 19
E:bearing.set/ps.set: PRODAT parameter file for colour dir grams 19
bearing.set 20
ps.set 20
F:Installation of Windows NT Tools 20
eXceed/XDK 6.0 20
Cygnus GNUWin32 21
Microsoft Visual C++ 5.0 21
Digital Visual Fortran 5.0 21

INTRODUCTION

In October of 1998, DREA had a requirement to add some functionality to PRODAT, as well as fix some discovered bugs in this signal processing program. PRIOR Data Sciences was contracted to perform these enhancements and corrections.

The work was broken down into four sections: [Task 1] Work for the Ocean Acoustics group, consisting of determining the cause of and correcting the loss of performance in newer versions of PRODAT, and later, verifying the capability to compile the NT Port of PRODAT [Nickerson 98]; [Task 2] Addition of the Harmonic Mean Normalizer [Cox 89] designed by Cox and Pace; [Task 3] Correct some problems with short FFTs and colour printing; and [Task 4] Add a new plot frequency-expansion procedure for improved visibility in grams.

This report is designed to serve the needs of several audiences. It shall report on the nature and status of the work that was performed for DREA. It shall acquaint the user of PRODAT with the usage of new features added to PRODAT. It shall acquaint the more technical user of PRODAT with the location of source code, give a cursorary introduction to the revision control system in which changes to PRODAT are stored, and procedures for compiling and running PRODAT. It shall aquaint the programmer tasked with modifications to PRODAT with the nature of modifications performed on PRODAT. Finally, it shall introduce a number of issues that appeared during the course of the development.

COMPILING PRODAT

REVISION CONTROL - CVS

The source code for PRODAT has been stored in CVS (Concurrent Version System), a GNU product. The CVS "Repository" is currently stored on Rainbow in the directory */usr/people/maxwell/CVSROOT*. Should the maxwell account contents be deleted, care should be taken to move this directory to a new place.

CVS can be used to perform several tasks. All CVS commands are accessed through the operating system command *cvs*, which is found in directory */usr/people/maxwell/bin* on Rainbow. The enviroment variable *CVSROOT* must be set to the directory of the Repository, or the command line option "-d <repository>" to CVS must be used.

COMPILATION

To obtain a copy of PRODAT for "distribution", that is placing on a machine to compile, one should issue the command:

```
$ cvs export -rRELEASE_322151 src/Prodat
```

This exports the contents of the state of PRODAT when "released" at the end of this contract. (322151 is PRIOR's project number for this contract.) One can also use a date specification such as *"-D20-Jan-1999"* to specify that one wants the state of the program as it existed on a particular day, in this case the 20th of January, 1999.)

This will produce a directory src/Prodat, containing all the source files needed to compile PRODAT. One can then package this directory with tar and transport it to the target machine. Once unpackaged on the target (either use the shell scripts *unpack (Unix/NT) / unpack.com (VMS)* or the command *tar/vmstar*), small modifications must be made to inform the *make* command which platform it is being compiled on. Follow the instructions in the file *read.me*, which instructs the user on what modifications must be made to machine.typ in order to compile correctly. When the machine.typ is correct, use the command *make (Unix), nmake (NT), or @make.com (VMS)* to actually do the compilation. This should produce the executables "SDU_Server", and "Prodat", as well as "pfe" on some platforms.

MODIFICATION

If a change is to be made to the source, then a "controlled" copy must be obtained with the command

```
$ cvs checkout src/Prodat
```

This will again produce a directory src/Prodat with all the sources which can be compiled in. However, since revision control information is also produced, changes can be recorded back into the CVS repository. After making a modification to the program one "commits" the changes. If several files were changed then one commits the changes for the directory, or lists the files on the command line. The revision comment can be recorded in all to indicate that the changes to all the files were done to introduce a new feature, correct one bug, etc. The command to commit changes is

```
$ cvs commit [files]
```

The files argument is optional and if omitted the command will be run for the current directory and its subdirectories.

OTHER FEATURES

CVS has many more features such as "tagging" (the facility used to allow exporting the "RELEASE_322151" version), "branching" (allowing one to introduce experimental features, and commit them without altering the main stable tree), features to facilitate concurrent development by several developers (eg update, merge), and comparison between versions (diff) to determine sources of instability or simply what has changed (and presumably why). For a further exploration of the usage and features of CVS, the reader is referred to the "INFO" page found in the info tree of a system installed version of CVS (such on the NGSP-IPE machine) or the */usr/people/maxwell/info* directory on Rainbow.

USAGE

LOCATION OF EXECUTABLES

The process of exporting and compiling has been performed for each of several platforms. The location of these executables is as follows:

Platform	Location
SGI (Rainbow)	/usr/people/maxwell/Current_Prodat/prodat.sh
MAXION (Viking)	/usr/people/maxwell/Current_Prodat/prodat.sh
VMS (FIRM)	ps_115:[maxwell.current_prodat]prodat.com
WindowsNT (BRIG)	D:\PRODAT\PRODAT.BAT

PRODAT was not compiled for the MASSCOMP, nor has a Solaris executable been generated.

To run the program, simply run the script file mentioned in the above table, which will set the path correctly and invoke the executable file in the same directory. On Viking, it will also set the DISPLAY environment variable to the second monitor.

NEW GRAMMAR ITEMS

In the course of modifications, some new commands were added to access new features. These new items are sumarized in the following table, and explained in detail under the usage section for the function that they invoke.

From Main Menu	SubCommand	Parameters
TSERIES	MULTI_DAT	ON
		OFF
NORMALIZE	HARMONIC	gap(#) noise_window(#) averager(PA/SWA) m(#)
GREY	EXPAND	factor

USAGE OF MULTIPLE .DAT FILE PINGS

Part of the work performed for the Ocean Acoustics Group was to augment the ping processing abilities [Murray 98] to add the ability to dump the data from a ping into a DAT file in addition to processing it. This is turned on through use of the *MULTI_DAT* command in the *TSERIES* menu. To turn this feature on, (from the main menu) use the command

```
TSERIES MULTI_DAT ON
```

To turn this feature off again, use `TSERIES MULTI_DAT OFF`.

USAGE OF HARMONIC MEAN NORMALIZER

A new normalizer was added. This normalizer is accessed like any other normalizer from the main menu with the *NORMALIZE* command. The Harmonic Mean (Cox) [Cox 89] normalizer is chosen with the *HARMONIC* subcommand.

Unlike other normalizers, it takes more parameters. Upon choosing the Harmonic Mean normalizer, the user will be prompted for the values of the various parameters which are (in order of request): gap width, noise window width, averager type, and M or history size. If one wished to configure the normalizer with a gap of 5 bins, a noise window of 40 bins, using the Progressive Averager with a history of 10, one would use the command:

```
NORM HARM 5 40 PA 10
```

It should be noted when entering the noise window width that this is in bins of signal, whereas the OTA and Split Window normalizers collapse the signal into a vector one-fourth the size of the original before doing their calculations then expand it out. Hence the noise window size chosen for the Harmonic Mean Normalizer must be 4 times that of the one that would be chosen for a Split Window normalizer.

USAGE OF GREYSCALE EXPANSION

A routine to widen the display of greyscales on the screen was added to increase visibility with small FFT sizes. This is invoked by entering the *GREY* menu and choosing the *EXPAND* command. One then enters the factor to expand by, be it 2 or 4. To expand the display by 2 (2 dots for every signal bin), use

```
GREY EXPAND 2
```

USAGE OF TIMESERIES DISPLAY

During the work with Multiple-DAT ping processing, the timeseries display code was observed. It was discovered that the keys to control the display had become lost in the mists of time. Therefore the keys available are listed here:

Key	Action
I	Zoom viewport IN (if display was -5V to 5V, then becomes-2.5V to 2.5V.)
O	Zoom viewport OUT
U	Move viewport UP (if display was -3V to 3V, then becomes-2V to 4V
D	Move viewport DOWN
R	RECENTER around 0.
X	eXit a started aquisition, or reset excursion count (untrigger)
S	START an aquisition manually (manual triggering)
N	Halt gram display like buffers had become overfull

TASK 1: WORK FOR OCEAN ACOUSTICS GROUP

The initial task was to identify and overcome software problems that were limitting the processing speed of the MAXION data aquisition system. When the problem could not be reproduced, Doug Caldwell requested that the PING processing be altered slightly to produce DAT files containing only the data values for the ping processing ("Multi-DAT Pings"). During the demonstration of this new functionality on the Windows NT machine, it was requested that some time be spent improving the documentation for and demonstrating the Windows NT port of PRODAT.

The approach taken in attempting to find the source and scope of the problem causing the loss of performance was to pick a consistent set of parameters, and run the various available versions of PRODAT with this set of parameters. The parameter file can be found in Appendix A. Timing tests were performed on five versions of the program from various stages of development. These results are summarized below:

Program	FFT-50	Aver	FFT-100	Aver	FFT-150	Aver	80%
pro_older	.085	.796	.071	.809			<200
pro_old	.077	.867	.065	.833	.098	.885	360
prodat	.083	.809	.083	.816	.072	.833	190
pro_new	.082	.836	.070	.836	.071	.848	210
pro_ping	.079	.847	.067	.874	.078	.883	290
pro_cur	.084	.825	.072	.823	.083	.830	290

In explanation of the table, the fields are as follows:

- **FFT-50** This is time per FFT at FFT number 50. Lower numbers are better (faster).
- **FFT-100** This is time per FFT at FFT number 100.
- **FFT-150** This is time per FFT at FFT number 150.
- **Average** This is the average speed of processing at the FFT number beside the column. Higher numbers are better, with 1.0 meaning that PRODAT is processing in real-time.
- **80%** The FFT number at which 80% of the buffers are consumed, and gram display is turned off.

The various vintages of the program are named by the command used to invoke them. All versions can be found in ~murray/Current_Prodat/{version} on the MAXXION machine *Viking*. The versions are as follows:

- **pro_older** An interum version as development continued.
- **pro_old** A newer interum version.
- **prodat]** This version is Release 3.0 22-SEP-97.
- **pro_new]** This is Release 3.0 15-OCT-97.
- **pro_ping]** This is Release 3.2 5-MAR-98, after the integration of changes for the Windows NT port, and PING processing, the current version before any changes made under this contract.
- **pro_cur]** This is Release 3.3b 19-JAN-98, the version that has been produced by this contract.

While the best performance was achieved with prodat_old, this version is an interum version, and not guaranteed to be correct. Since this version, the performance has in fact been becoming progressively better, culminating in pro_ping.

During the testing, two bugs were discovered. The first was that the screen manipulation code did not seem to be clearing the screen properly. The second was that if the overlap value was changed from 50 to 25 after executing the parameter file from Appendix A, the GRAM_ON parameter seems to be set to FALSE, resulting in grams not being displayed.

When the first bug was investigated, it turned out to be more insidious that was apparent at first glance. The problem resulted from the command table size not being raised in SASIO during the last command set additions. When the command table was loaded, it was overwriting sections of the screen manipulation codes table. This problem has been corrected, but two things must be noted from this: First, when the command set is augmented it is often necessary to augment the size of the command table. Second, in attempting to raise the size of the command table, difficulties were encountered in getting GENCOM (the SASIO program that generates compiled SASIO command tables) to function correctly on the SGI. It works now but it is unclear whether this is reproducable when the command table size must be raised again.

With regard to the second error where the value of a parameter is set one way, then altered later, this could probably be fixed, but it is symptomatic of difficulty in keeping track of the effects of old parameters on newer parameters, which affect even newer parameters.

Finally, it is observed that the latest prodat has become slower. However, the number of FFTs before the buffers become filled is unchanged from prodat_ping. In light of this, it was decided to leave this question until it becomes a problem.

MULTI-DAT PINGS

When the loss of performance could not be reproduced, it was decided that the ability to generate multiple DAT files for pings should be added to PRODAT.

In early 1998, ping processing was added to PRODAT [Murray 98]. This allows PRODAT to be triggered to process and display data based on the signal making a specified number of excursions above a certain strength. The system then becomes "triggered". It would start processing the data starting PRE_TIME seconds before the detection and proceeding for PING_TIME seconds. This PING_TIME seconds of data is the "ping window". The results of processing would be displayed on the screen and would be stored in a PWR file.

It has since been discovered that it would also be convenient for PRODAT to be able to "snip" the ping out of the data file, and store it in a separate DAT file containing only the ping window. That which once was only processed into a PWR file, is now also stored in a DAT file. It is hoped that this will make it easier to reprocess a specific ping at a future time, rather than having to skip through the entire file or tape to get to the correct place.

This functionality has been implemented, and is available from the *TSERIES* menu where the other ping processing commands are found. This functionality is turned on with the command **`MULTI_DAT ON`**, and turned off with **`MULTI_DAT OFF`**. The filename for the series of DAT files is constructed by taking basename of the PNG file, and appending "_pNNN.dat" to this, where NNN is the ping number. For example, if the PNG file were named "ping_trial.png", then the DAT files would be "ping_trial_p001.dat", "ping_trial_p002.dat", and so on.

IMPLEMENTATION

The sequence of PING.DAT files is started by the PING_DATF_OPEN subroutine called from INIT_TS (ts_plot.f) if the PING_MULTIFILE logical variable is set to true. This opens the first file in the series and writes an appropriate header. No data is written until the ping is triggered.

When a ping is triggered, PRE_TIME seconds of data is dumped by the PING_DATF_DUMP subroutine, called from TS_CHANNELS (ts_plot.f). Then as data becomes available to be processed, it is dumped again using PING_DATF_DUMP.

At the end of PING_TIME seconds when the system becomes untriggered, the subroutine PING_DATF_NEXT is called to close the current PING.DAT file (rewriting the header with updated details of number of data samples, etc.), and opens the next PING.DAT file, writing the header, then waiting for the trigger.

After all the data has been monitored and processed or not, the sequence of files is closed with the PING_DATF_CLOSE subroutine called from the DEINIT_TS subroutine.

WORK WITH WINDOWS NT PORT

While verifying that the port still works on the Windows NT machines in November, it was decided to expand the task to verifying that PRODAT can be reproducably built on an NT machine.

The machine was taken back to having only Windows NT and FX32 (ix86 emulator software) installed on it. The following packages were reinstalled: Microsoft Visual C++, Digital Visual Fortran, eXceed/XDK,
and GNUWin32. PRODAT was then copied from the current version on Rainbow, and compiled according to the instructions. Detailed notes on how to install each of these packages can be found in Appendix F. This appendix is designed as an addendum to the report on the port of PRODAT to Windows NT [Nickerson 98].

Once PRODAT was successfully compiled, several functions were demonstrated and exercised. Among these functions was the ability to acquire over the network. This did not initially work, and some modifications had to be made. The IP address that was hard-coded into network.c had become incorrect since the application was written. Code to determine the IP address of the interface was uncommented, and checked. It appears to be working at this time, but if it fails to work in the future, the hardcoded IP address can be changed in network.h.

PRODAT FRONT END

The Prodat Front End was written during the port to Windows NT to provide a GUI front end to PRODAT, hopefully an easier way of using PRODAT.

During this contract, the PFE was ported to Solaris and Irix (which should work on any other Unix platform with X11R5 and Motif 1.2 installed). In addition, several more commands were added.

It was found that from a programmer's standpoint, it was not documented as how new commands are added. This task is accomplished first by finding the menu into which the command is to be added. In *pfe.c*, the menu structure is laid out in a series of static menuItem structures. The main menu points to the file menu, which points to the "Input Source" menu, which has a "Data File" item, which is linked to the infile_cb callback. Likewise each of the other menus. Each submenu is a menu item on the main menu, a submenu can have items that go directly to a callback and items that go to a submenu. Callbacks might create the dialog box necessary to get a text or numeric input, or might simply process which menu item was selected to alter the global structure that remembers the state of PRODAT so that reasonable bounds can be set on input data. Dialogs created must have callbacks to process them. Examples of how this is done abound in the code. It is suggested that to add new dialogs, simply find a similar dialog, and modify the code appropriately.

TASK 2: HARMONIC MEAN NORMALIZER

The next task was to add the normalizer discussed in the the paper by Cox and Pace [Cox 89] describing a better normalizer. Two normalizers are already available in PRODAT, the Split Window Normalizer and the OTA Normalizer. The Cox Normalizer, or Harmonic Mean Normalizer is based on not only comparing a bin to its neighbours, but also to the strength in that bin in the past. Since the other normalizers operate only on the particular spectrum fed to them, it was necessary to add code to collect a history of values. As a result of this, it is suitable to call this normalizer only if calls are made to collect historical values for the spectrum being normalized.

A Normalizer operates by taking the average signal in a noise window on both sides of the bin of interest to get a noise estimate, and adjusting the value in that bin based on the result of this calculation to accentuate signals and reduce the effect of noise. A gap is left between the bin of interest and the noise window so as not to factor in signal that has leaked out of the bin. Usually this average is the arithmatic mean, and usually it is calculated based only on the current sample. The Cox normalizer attempts to improve the calculated noise estimate at only a small increase in calculation by using the *Harmonic Mean*, and including historical values in the calculation.

The exact steps in the calculation are as follows:

1. Average M (input from prodat.f) samples of raw spectral data. A "recursive average" is used. (See discussion of averages below.)
2. Invert the Values to obtain y_i^{-1}.
3. Form reciprocal of harmonic mean H_j^{-1} for the noise window associated with each cell.

$$H_j^{-1} = \frac{1}{N} \sum_{i=1}^{N/2} y_{j+g+i}^{-1} + y_{j-g-i}^{-1}$$

ie Sum over N/2 numbers each from the left and right hand sides of a gap away from the cell of interest. This is in fact done by adding the entire vector to itself offset by 1 element, window/2 times.

4. Optionally, average K scans of raw spectral data to obtain improved estimates of signal-plus-noise x_j.
5. Form the normalized cell values by

$$z_j = x_j\left(\frac{M-1}{M}\right)H_j^{-1}$$

Note that the use of the *Harmonic Mean* (sum of inverse of samples times number of samples) means that small numbers will dominate in the noise estimation. Large numbers are presumably signal, hence it is reasonable to expect this approach would be very effective.

The user of this normalizer should take note that several implementation decisions were made. Two different definitions of "recursive average" are offered for usage (see below.) Also note that unlike the other normalizers currently in PRODAT, the data to be normalized is not compressed down by a factor of 4 before being operated upon. Hence if the user were using the default values of gap size 2, noise window size 20 from other normalizers, when invoking this normalizer they should use a gap of 8 and a noise window size of 80.

Finally it should be noted that while the other normalizers act on the spectrum after it has been converted to decibels, the Harmonic Mean Normalizer must be invoked on the raw spectrum. The lower power values in a spectrum will be small numbers in the raw spectrum, but will be large negative values on a logarithmic scale. If a strong signal has a value of 50 dB, a medium signal has a value of 0.5 dB and noise -120 dB, inverting these numbers and adding them to similar numbers will tend to accentuate the medium signal over both the noise and strong signal, which is not the desired effect. Therefore the raw spectral values are operated on instead. This required some extra complexity as the call to the function which puts the spectrum onto a decibel scale (VDBPWR) is also used to scale the numbers by using the scale factor as the reference number. Some of the factors are scale factors (eg RFFT must have its output scaled by 1/2N to get correct values) so must be applied before applying the normalizer. Others are not scaling factors, but are supposed to be applied to the number from the accumulator to achieve the averaging of K spectrum scans. This is appropriate when the actual normalizer function is invoked, but not when collecting the history.

On a side note, as a debugging tool, facilities were added to allow the display of the spectrum about to be greyscaled (that which was just normalized). This is turned on by changing the value of the variable IDOGRYSPECT to 1 in prodat.f and recompiling the program. If this is considered to be a useful tool for general operations, it would probably not be difficult to add a command to turn this on at runtime.

DISCUSSION ON AVERAGERS

In the course of reading the paper on Harmonic Mean Normalization, the term "Recursive Average" was encountered. Several sources both literary and oral were consulted, but no definitive definition was found. The paper [Cox 89] mentions this term as follows:

> 1. Average M scans of raw spectral data to obtain improved spectral estimates y_i. A *recursive average* may be used.

As well, as in the example usage of the normalizer (section 4.3):

> ....The following normalizer parameters were used: K = 1, g = 3, N = 32, M ≈ 19 (*recursive average* with $\alpha = 0.1$).

One source postulated that is was simply a synonym for "running average", that is y_i is the mean of the last M scans. This required the allocation of a new block of memory to store the last M raw spectra, and a tap into the main processing loop to collect these scans as they went by.

Another definition that the author has advanced is to allocate an accumulating average which would be added to in the following way:

$$average_j(i) = \left(average_j(i-1) * \frac{M}{M-1} \right) + \frac{scan_j}{M}$$

that is, weight the average toward the "prototypical" value for that cell from the past, and add on the latest scan for that cell. If noise is considered to be a constant (unvarying) function in the background for a sampling session, for which we are trying to determine an estimate, then it is believed that this method has merit, since it does not throw away any old samples, simply refines the result with newer data. A larger M causes this function to alter more slowly making it less affected by transient alterations or the sudden presence of a signal. A smaller M makes the function more dynamic, either converging faster with price of sensitivity to transient effects, such as a signal, or more responsive to the actual noise function actually being dynamic or variant over to course of a sampling run.

In terms of implementation, the advantage of the "progressive average" (PA) method has a memory savings in having to allocate only the amount of memory needed for one scan line. In both cases a tap is needed into the main processing loop for each scanline, but both could be placed in the current averaging code. In the "Running Average" (or Sliding-Window Average (SWA)) method, the memory access and allocation routines were isolated in the normalizer code.

It has been asked that any variables that could be variable be configurable, as this is an experiment in normalization. In light of this being an experiment, one of the configurable parameters is which averaging method is used. Obviously, both methods will increase processing time slightly above the time required for either Split Window or OTA normalization.

IMPLEMENTATION

The normalizer was implemented in C to allow for dynamic allocation of the history array. Fortran access routines were created to allow these functions to be called from process.f.

An extended version of the NPARMS subroutine (FEXT_NPARMS) was written to handle the extra parameters needed by the harmonic mean normalizer (such as history size). The parameters to the normalizer are set up by this routine called from prodat.f. Each time a spectrum is produced it is stored in the history by the NORMAVG_SAVE routine. In the case of the progressive average this calculates a new average; with sliding window average, the line is simply stored in a circular buffer for calculation of the average during the normalization.

When the normalizer (FNORMALIZE, a front end to harmm_norm) is actually invoked, the history is plucked from the history array, and the harmonic mean of the noise window is calculated for each bin according to the above algorithm (edge bins are calculated on a best effort, but can not be considered to be as good an estimate as center bins.) This is then applied to the signal passed into the routine.

TASK 3: CORRECT PROBLEMS WITH SHORT FFTS

Recently, DREA has begun to use PRODAT with shorter FFTs with the shift from passive to active sonar work. When the FFT size was set to 128 points or less, it was discovered that normalization stopped working correctly. In addition, it was discovered that the printing of Colour-Direction grams did not work correctly.

NORMALIZATION

PRIOR was asked to investigate why the normalization procedures did not work correctly at FFT sizes of less than 128 points. After a bit of investigation, it was discovered that the default noise window size is 20. Since the OTA and Split Window normalization methods divide the number of data points by 4, this is an effective window size of 80. A 128-point FFT produces 64 points, making the noise window larger than the number of available points. This has been repaired by inserting a check that the noise window is not larger than than one-quarter of the number of frequency bins output by the FFT. (One-quarter because of the 4-1 compression of the data to be normalized performed before the process begins.) The Split Window and OTA normalizers can therefore not handle FFT sizes below 32 (2 noise window bins and 2 gap bins on each side of the target bin maps to 32 data bins, which is larger than the 16 spectral points from such an FFT.) In theory the Harmonic Mean Normalizer can handle the smaller FFTs better, but seems to perform badly at 32 points also. Therefore the default is to turn normalization off completely if the FFT size is below 32 points.

This was tested with the parameters in task3.set, and task34.set, found in Appendix D.

COLOUR PRINTOUTS

It has been discovered that colour-direction grams lose features when printed. It appeared that the display had more contrast than the printout.

The approach taken to correcting the problem was to follow the data from its generation through the two possible paths it could take to display. The data is generated by the *BEARING* function. It is then dumped in the form of (intensity, direction) pairs to a BRG file, or it is *NORMALIZE*d, *GREYLEVEL*ed, then given to *display_colordir_grams* for screen display. From a BRG file, the hardcopy code reads the (intensity, direction) pairs, *vnormalizes*, then *grey_levels* them. The result of these two parallel data paths is a set of greylevelled values. It was verified that these two paths produced the same values, before the colour mapping code was analyzed. It was discovered that a small error was present in that the default normalization (Split Window, 20 points) was recorded in the BRG file if normalization was turned off. This was corrected, and the grey levels were the same.

Next the two colour mapping functions were compared, and differences were discovered. The display routine utilizes a linear scale for the alteration that intensity has on the colour used (the intensity is multipled by the base colour values. The printout code used a fourth-root scale (the fourth root of the intensity is multiplied by the base colour (Red-Green-Blue) values). The same base colours are used. Obviously this reduces contrast between the colour chosen for the maximum and minimum intensities.

Several printers were used to test the appearance of the printout: The HP ColorLaserJet5 in Room 335, the Tektronix printer in Ian Fraser's office, and Ghostview were all used. The largest problem with contrast was discovered on the HP. In fact, with the original colours, the blues representing North-East and North-West were indistinguishable.

After some experiment it was discovered that a combination of approaches achieved the desired effect of making the features visible on the screen equally visible on the printout. First, the blues for NE and NW were separated further by adding more Red and Green components, until there was a visible difference. Next the fourth root scale was dropped in favour of a linear scale. Finally, the colours were "washed" [1] to make them paler, allowing a greater range before the intensity causes the colour to appear very dark. This effect has achieved approval even when printed on the HP.

However, it should be noted that in the course of generating test pages, and such, a comparison postscript file was discovered in the murray account on Rainbow. This file, "test_sq.ps", showed a sample plot, which is presumably the test case for the original addition. In this example there were features on a sea of black in the linear scale, but features with some other varying background with the fourth-root scale. This time, the test was performed with the data file "jl141593.dat", the command macro file "bearing.set" to generate the BRG file, and "ps.set" to generate the Postscript file. Both SET files can be found in Appendix E.

Because it is assumed that there is a reason for the previous choice of colours, this colour scale is still available by calling the place_grams function (hardcopy.c) with the COLOUR_BEARING_4R_VIVID constant as the colour scheme. There is currently no way to access this from the HARDCOPY menu, since it was postulated that this might confuse the issue. If users find that the new colour scheme is inferior to the old for some plots and not for others, then it should be trivial to add the facility to print in either chosen at runtime.

TASK 4: WIDER GRAMS

Smaller FFTs mean fewer frequency bins to display on the screen. Fewer frequency bins means that features that might have spanned several bins, and so several screen pixels are only in one bin, and so are harder to see. It was requested that a routine be written to expand the width of features just as there is a routine to compress displays if they are too wide for the screen.

This task was accomplished by creating a routine, VINTERPOLATE, that takes as input a spectrum and expands its width by a factor of two to four by interpolating values between each existing bin. The setup routine for greyscale display was informed that it will be given twice as many frequency bins to display, and VINTERPOLATE was called on the spectrum before GREYLEVELing it then passing it to display_greyscales. (Another routine, EXPAND_GREY, was written which simply copies each greylevelled value into 2-4 bins before display_greyscales, but linear interpolation was considered to be a much more aesthetic solution.)

[1] Adding more strength in components that once were weak. For example, to wash Pure Red (1.0, 0.0, 0.0), one would choose the values (1.0, 0.4, 0.4).

MISCELLANEOUS ISSUES

A number of miscellaneous issues came up during development, and they are discussed below.

MAXION MEMORY

On the MAXION platform, memory seemed to be somewhat more scarse than on other platforms. On several occassions PRODAT stopped running due to lack of memory. The first occassion spontaneously disappeared after several reboots. The second occassion was indeed because the PRODAT image had grown beyond the bounds of feasibility. This latter problem was delayed by going through the process.f file making a number of large "temp" local arrays AUTOMATIC (not STATIC, not SAVE) variables. As a result they are allocated only if the routine is used and deallocated when it leaves the routine. Although these arrays only appear to total one to two megabytes in size, this shaved nine megabytes off the size of the running image, and allowed PRODAT to fork the SDU_Server process for screen display.

PRODAT uses the -static compiler flag because many older routines are written assuming the VAXFortran behavior of defaulting all local variables to SAVE. Since the routines that make this assumption are not precisely identified, it is not considered safe to do this for all files unless they have been checked to verify that this is not a problem.

On Unix ports, the SDU_Server is spawned at present with the use of the "system" library call. This does not return any sign of failure if the program is unable to launch the SDU server. A quick attempt was made to rewrite this routine with a sequence of "fork" and "exec" library calls, but this proved unsuccessful as the SIGINT signal (Control-C) was being sent to both the SDU server and the PRODAT program when a request was made to stop aquisition on the MAXION. While it is possible to alter the signal mask on the child process to not catch these signal just as the system library call does, this is one of the areas that Unix flavours differ, so portably doing this would be more than a moments work. It was therefore decided to leave the code in the way that it works in the standard case and revisit the question if PRODAT grows to excessive proportions again.

GENCOM/SASIO

As mentioned in the section on Task 1, early problems were encountered with regenerating a new command table using GENCOM on the SGI and MAXION platforms. (Previously this was done on the MASSCOMP platform.) This started working suddenly and it is unclear exactly what the required sequence of actions to increase the table sizes is. The list of actions taken was:

1. Increase MAXTAB parameter in GENCOM.F
2. Increase MAXTAB parameter in the output code of GENCOM.F.
3. Increase MAXTAB parameter in COM.FOR
4. Regenerate the GENCOM_TAB.F using GENCOM, and place the resulting COM_TABLE in the GENCOM.F file, and recompile.
5. Regenerate the PRODAT_TAB.F file.

Which of these actions were necessary, and what the correct order was is unclear. GENCOM can currently generate correct PRODAT_TAB.F's and will continue to do so until it becomes necessary to add 4 more command tables to the PRODAT.TAB file. (MAXTAB would have been increased by a bigger factor, but taking a conservative approach it was decided to start small, and once it was working, there was
significant hesitation to altering it for fear of breaking it again.)

BUFSIZ

Very near the end of the contract, it was discovered that the BUFSIZ parameter in PARAMETERS.COMMON was not set correctly for some tapes. During the port to Windows NT [Nickerson 98] it was discovered that in order to read tapes from the VME bus machine, it was necessary to have this parameter set to 32768 (32K). This did not appear to cause problems reading other tapes, but it has now been discovered that in order to read tapes from the IMPACT system (DLT tape "May 31, 1998; CAN/OK Flight 2; 20:18 →") this parameter must be set in the range 20580 to 21400 (between 20K and 21K, excluding exactly 20K and 21K). Obviously this is very strange and not acceptable, but until this can receive a detailed analysis on the VMS platform the value is set to 21400 and other platforms are set to 32768. It is believed that this is only a problem with acquisition from tape, not disk.

MISCELLANEOUS FIXES

It was discovered in March, 1998 that PRODAT was unable to skip forward by more than 32 kilo-tapeblocks on the VMS platform due to a bug in the library function invoked to perform this action. Also it was discovered that if a tapes starts at say 23:00 and spans 4 hours of data, setting START_TIME to 01:00 resulted in attempts to skip past the beginning of the tape.

Patches for both these problems have finally been integrated into PRODAT.

CONTRACTOR ACCOUNTS

It should be noted that CSG has a policy of deleting contractor accounts. If this is to happen, then it will be necessary move the files mentioned in this report to a safe place that will not be deleted. Important locations are as follows:

- **rainbow:/usr/people/maxwell/CVSROOT** CVS Repository containing the latest code.
- **rainbow:/usr/people/maxwell/Current_Prodat** The current prodat code and executable for SGI.
- **rainbow:/usr/people/maxwell/bin** Contains the cvs executable for accessing this repository
- **firm::ps_115:[maxwell.current_prodat]** Contains current prodat and executable for VMS.
- **rainbow:/usr/people/murray** Kent Murray's account contains the code and executables for everything he has worked on.
- **firm::ps_105:[murray]** Likewise there may be useful things for things worked on under VMS.

CONCLUSIONS

During this contract several bugs were fixed including problems with short FFTs and problems with Colour-Direction gram printing. Several new features were implemented including the Cox, or Harmonic Mean Normalizer; a facility for expanding greyscale displays; and the ability to generate per-ping DAT files from ping processing. A problem with newer versions of PRODAT being slower than older versions on the MAXION was investigated, but could not be reproduced. The ability to compile PRODAT on Windows NT was verified and the process documented. Finally a number of other small issues were identified.

BIBLIOGRAPHY

[Cox 89] H. Cox, D. Pace, "A Fast Normalizer", Proceedings of the 30th Asilomar Conference on Signals, Systems, and Computers, 3-6 November 1996, IEEE Computer Society Press.

[Murray 98] K. Murray, "Active Sonar Additions for PRODAT", DREA Contractor Report, 98/DRAFT (1998).

[Nickerson 98] J. Nickerson and S. Osgood, "Port of PRODAT to the Alpha-Windows NT Platform", DREA Contractor Report 98/DRAFT (1998).

[Osgood 95] S. Osgood and K. Murray, "The development of tools for analyzing narrowband ambient noise", DREA Contractor Report 95/438 (May 1995).

APPENDICES

A:TEST26_98.SET: PRODAT PARAMETER FILE FOR TASK 1

This SET file is used to provide a consistent set of parameters for testing the speed of processing to determine the slowdown observed in processing on the MAXION between successive versions of PRODAT.

```
inf dacp
freq 5120
start 04:15:00
time 00:55:10
speed 1
Date 10-May-98
chan 1 2 3 4 5 6 7 8 9 10 11 12

out none

contig off
des
  1  4  954D om
  2  4  954D sin
  3  4  954D cos
  4  5  954D om
  5  5  954D sin
  6  5  954D cos
  7 17  53D2 om
  8 17  53D2 sin
  9 17  53D2 cos
 10 18  53D2 om
 11 18  53D2 sin
 12 18  53D2 cos

ex
fft 8192
over 50
wind hann
ave fft 80
chan 1 2 3 4 5 6 7 8 9 10 11 12

grey
greych -1

sbin freq 0 1000
scravg 4
scrchan 1 2 3 4 5 6 7 8 9 10 11 12

spect on
spax log
spset 50 100
spbin freq 1 2000
spwind  1 1 4 1 7 1 10 1 2 2 3 2 5 2 6 2 8 2 9 2 11 2 12 2
```

```
ex
cal on
set DIFAR_Q53D.187 0 1 4 7 10

set DIFAR_Q53D.187 4 2 3 5 6 8 9 11 12

ex
```

B:SET FILES TESTING MULTI-DAT PINGS

Apr24_1P.set is used to generate the PING.DAT files.
Apr24_1P-jp.set displays the output for just ping number 2.

APR24_1P.SET

```
inf Apr24_1P.dat
fft 2048
over 50
wind hann
ave fft 4
chan 1 ^

ts multi_trig on
  multi_dat on
  multi_time 2.0
  pre_time 1.0
  multi_proc pulse
  tchan 1 ^

  excursion 2
  thresh 0.25

  on
  chan 1 ^

  level -2 2
  time 10
 exit
```

APR24_1P-JP.SET

```
inf Apr24_1P_p002.dat
fft 2048
over 50
wind hann
ave fft 4
chan 1 ^

ts
```

```
on
chan 1 ^

level -2 2
time 10
exit
```

C:SET FILES FOR HARMONIC MEAN NORMALIZER TESTING

HARMM.SET

```
inf directed.dat
over 0
aver fft 4
set set 15
grey
  spect on
  scrch 1 4 9 12 ^
  spcha 1 4 9 12 ^
  spset -120 40
  exit


norm
  harm
  3 32 PA 10
```

D:SET FILES FOR TASKS 3 AND 4

These SET files simply test the processing at short FFTs, test expanding greyscales, and expanding greyscales with short FFTs.

TASK3.SET

```
inf c32_f2000_t05.dat
over 0
fft_size 128
aver fft 32
set set 15
grey
  spect on
  scrch 1 4 9 12 ^
  spcha 1 4 9 12 ^
  spset -120 40
  scravg 4
  exit
```

```
norm
  split  8
```

TASK4.SET

```
inf test64.dat
start delta 00:01:00.01
set time 00:01:00
aver fft 5
grey
   spect on
   exit


fft 512
aver fft 10
set time 00:00:30
grey
  expand 2
  exit
run
```

TASK34.SET

```
inf c32_f2000_t05.dat
over 0
fft_size 128
aver fft 32
set set 5
grey
  spect on
  scrch 1 4 9 12 ^
  spcha 1 4 9 12 ^
  spset -120 40
  scravg 4
  expand 4
  exit


norm
  split  8
```

E:BEARING.SET/PS.SET: PRODAT PARAMETER FILE FOR COLOUR DIR GRAMS

This is the parameter (SET) file used to tune the colours for hardcopy printing of bearing (BRG) files.

BEARING.SET

```
INF jl181659.dat
FFT 128
OVER 0
AVE FFT 2
NORM NONE
start time 17:03:32
sets time 00:00:50
CROSS
  BEARING ON 3 1 2 ^ 20
  EXIT
GREY
  SET -18 16 2
  CD_ON
  CD_CHAN 1 ^
  EXIT
```

PS.SET

```
hardcopy
 bfile jl181659.brg
 laser off
 pfile bearing.ps
 print
 exit
quit
```

F: INSTALLATION OF WINDOWS NT TOOLS

EXCEED/XDK 6.0

1. Obtain eXceed/XDK 6.0 CD-ROM from Doug Caldwell's Desk
2. Insert in CD-ROM drive
3. Select Install XDK
4. In the Install Wizard, Choose Next
5. Choose Install Type: *Typical*
6. Allow Users to use?: *Yes*
7. Install Destination: *E:\ program files\Exceed.NT*
8. User Directory: *E:\program files\Exceed.NT\user* (the default)
9. Allow it to keep all selected components.
10. Fonts: *Misc, 100dpi, Andrew, DEC, HP, PC*
11. Choose the Finish button
12. Enter the serial number from the back of the CD Case, and enter the site contact as "Doug Caldwell".
13. Register: *Later*
14. Choose the Finish button.
15. Keyboard File: *US.KBF*
16. Enter the Configuration Password as Blank. (It can be changed later with XConfig.)
17. Put Shortcut on desktop: User Preference

18. Tune XServer?: *Yes*
19. Finish and Reboot.

CYGNUS GNUWIN32

1. Obtain from `http://www.cygnus.com/` → CygWin Button → Download Sites →`ftp://ftp.reversion.ca/pub/mirrors/cygwin/cygwin-b20/user.exe`
2. Go to Start → Settings → Control Panel → Add/Remove x86 Programs
3. Choose the Install Button, and enter the location of user.exe.
4. Instruct it to unpack to some temporary directory, such as c:\ temp.
5. A welcome screen will be displayed. Choose Next.
6. Accept the GNU Public License.
7. Install to D:\cygnus
8. Click the Finish button
9. It might be necessary to edit the path to add Cygwin:
10. Start → Control Panel → System,Environment Tab. Edit PATH to include D:\cygnus.

MICROSOFT VISUAL C++ 5.0

1. Obtain the MS Visual C++ 5.0 CD from Doug Caldwell's desk, and insert it in the CD-ROM drive.
2. Choose "Install Visual C++ 5.0 Professional"
3. Next, Accept the license agreement
4. Enter the Serial Number found on the back of the CD-ROM Case
5. Choose "Typical Installation" to E:\Program~Files\DevStudio
6. An information screen will appear, Choose Next button
7. Register the environment variables: Choose OK, and OK again.
8. Reboot

DIGITAL VISUAL FORTRAN 5.0

1. Obtain the Digital Visual Fortran 5.0 CD from Doug Caldwell's desk, and insert it into the CD-ROM drive
2. Click on Install Visual Fortran
3. Choose the Next button, then Choose OK on the license acceptance screen
4. Enter the serial number found on the back of the CD case
5. Choose a "Typical Install" to E: \Program~Files\DevStudio\
6. Group to insert in Start Menu?: *Next button*
7. Update Environment Variables: *Yes*
8. Click the Finish button
9. Reboot

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document.. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Prior Data Sciences Ltd
5475 Spring Garden Rd Suite 601

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

PRODAT Enhancement and Repair

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Maxwell, Chris

5. DATE OF PUBLICATION (month and year of publication of document)

May 1999

6a. NO .OF PAGES (total containing information Include Annexes, Appendices, etc).

21

6b. NO. OF REFS (total cited in document)

4

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

CONTRACTOR REPORT

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

Defence Research Establishment Atlantic
PO Box 1012, Dartmouth, NS, Canada B2Y 3Z7

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant).

Project 1cc11; 1cb13

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written).

W7707-8-5997

10a ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

10b OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

DREA CR 1999-048

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This report details work performed on the DREA data acquisition and analysis program PRODAT from Nov 1998 to Jan 1999. A Harmonic Mean Normalizer (Cox and Pace, Proc. 30th Asilomar Conference, Nov 96), a greyscale expansion routine, and a facility for generating multiple DAT files from ping processing were added. Problems with short FFTs and colour-direction sonogram printing were corrected. Loss of performance on the MAXION platform with newer versions of PRODAT was investigated.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

PRODAT
signal processing
spectrum analysis
spectrum display
greyscale